future internet

MDPI

Article

Sharing with Live Migration Energy Optimization Scheduler for Cloud Computing Data Centers

Samah Alshathri [1,*], Bogdan Ghita [2] and Nathan Clarke [2]

1 College of Information Technology, Princess Nourah Bint Abdulrahman University, Riyadh 11671 P.O.Box: 84428, Saudi Arabia

2 School of Computing, Electronics and Mathematics, University of Plymouth, Plymouth PL4 8AA, UK; bogdan.ghita@plymouth.ac.uk (B.G.); Nathan.clarke@plymouth.ac.uk (N.C.)

* Correspondence: sealshathry@pnu.edu.sa; Tel.: +966-11-823-8370

Received: 22 June 2018; Accepted: 25 July 2018; Published: 6 September 2018

Abstract: The cloud-computing concept has emerged as a powerful mechanism for data storage by providing a suitable platform for data centers. Recent studies show that the energy consumption of cloud computing systems is a key issue. Therefore, we should reduce the energy consumption to satisfy performance requirements, minimize power consumption, and maximize resource utilization. This paper introduces a novel algorithm that could allocate resources in a cloud-computing environment based on an energy optimization method called Sharing with Live Migration (SLM). In this scheduler, we used the Cloud-Sim toolkit to manage the usage of virtual machines (VMs) based on a novel algorithm that learns and predicts the similarity between the tasks, and then allocates each of them to a suitable VM. On the other hand, SLM satisfies the Quality of Services (QoS) constraints of the hosted applications by adopting a migration process. The experimental results show that the algorithm exhibits better performance, while saving power and minimizing the processing time. Therefore, the SLM algorithm demonstrates improved virtual machine efficiency and resource utilization compared to an adapted state-of-the-art algorithm for a similar problem.

Keywords: cloud computing; scheduling algorithm; green computing; energy optimization; virtualization; simulation

1. Introduction

Cloud computing represents the preferred alternative for on-demand computation and storage where clients can save, retrieve, and share any measure of data in the cloud [1]. Aside from their benefits, cloud-computing data centers are facing many problems, with high power consumption being one of the most significant ones [2]. In this context, a typical challenge faced by cloud service providers is balancing between minimizing energy usage and the delivered performance. Further, the current trend expects that the energy consumption of the United States' data centers will be about 73 billion kilo watts per hour by 2020 [3]. If power consumption continues to increase, power cost can easily overtake hardware cost by a large margin [4]. In the cloud data center, 90% of the electricity is consumed by the server, network, and cooling system [5].

As this industry is becoming a major resource and energy consumer, high efficiency cloud-computing data centers need to be developed in order to guarantee a green computing future [6] and meet the industrial needs. Scheduling techniques are designed to assign tasks in an optimal fashion for utilizing system resources [7]. Numerous scheduling models have been proposed and studied comprehensively, but most of them are unsuitable for the current cloud-computing environment, which raised the need for more efficient and innovative scheduling models [8].

Task scheduling is a key element in cloud computing, and currently focuses on all computing resources, so a scheduler could classify and analyze computing resources reasonably and reduce the execution time of tasks [9]. Optimizing the efficiency of the scheduling mechanism in a cloud environment can increase the performance of the server and associated resources, and maximize the processes managing proficiency [10].

Heuristic, hybrid, and energy-efficient task scheduling algorithms are the three main different categories of task scheduling [11]. To begin with, heuristic task scheduling can be divided into two types, namely static and dynamic power consumption. Static algorithms assume that all of the tasks are independent, and they all arrive at the same time. Many task-scheduling approaches categorized under the static scheduling methods such as minimum execution time and minimum completion time, min–min and max–min, genetic algorithm, etc. The dynamic scheduling techniques assume a dynamic nature of the tasks' arrival time and depend on the system machine's state of time, such as K-Percent and suffrage scheduling techniques.

The second class of algorithms regards energy efficient task scheduling. Task scheduling is one of the critical determinants in energy consumption for a cloud data center. Innovative green task-scheduling algorithms are designed and established to reduce energy consumption by determining the optimal processing speed to execute all tasks before a deadline. Finally, hybrid scheduling combines multiple scheduling parameters in one scheduling technique to decrease the execution time and produce a hybrid scheduling. Most of these algorithms are either novel or designed by combining one or more of the old predesigned schedulers.

The remainder of the paper is organized as follows. Section 2 presents related works. Section 3 builds the Sharing with Live Migration (SLM) model. Section 4 proposes the SLM scheduler algorithm. Section 5 presents the simulation and analysis. Section 6 presents the experimental Results. Finally, Section 7 addresses conclusions and future work.

2. Related Work

Many papers deal with task allocation and scheduling in a cloud environment, which successfully led to a reduction in the total energy consumption.

In [12], the authors proposed a Dynamic Voltage and Frequency Scaling (DVFS) enabled energy-efficient workflow task-scheduling algorithm (DEWTS). The proposed algorithm balances the performance of scheduling by dividing the parallel tasks in workflows, and then executes them at an appropriate time to reduce the power consumption. Evaluation tests indicated that the algorithm succeeded to drop the power consumption by 46.5% in the execution of parallel tasks.

The priority algorithm from [13] is a green energy-efficient scheduler that efficiently uses the DVFS technique to allocate proper resources to jobs according to the requirements of jobs under the Service Level Agreement (SLA). This technique improved the resource utilization and reduced the power consumption of servers by 23% in comparison to the dynamic voltage scaling technique [14]. An offline scheduling strategy for data intensive applications was proposed in [15] to reduce the energy consumption by degrading the SLA rate. The proposed scheduling strategy decreased network traffic and improved the utilization of servers to reduce the energy consumption in the cloud.

In [16], the authors introduced an energy aware scheduling strategy with DVFS integration in multicore processors, which works for batch and online mode tasks. The scheduler improved the energy consumption by 27% for batch mode applications, and 70% for the online mode applications. The proposed new virtual machine scheduler in [17] was used to schedule the virtual machines by selecting first the highest performance power virtual machine (VM) in the deadline constraint. This scheduling algorithm increased the processing capacity by 8% and saved up to 20% of the energy. Pavithra and Ranjana [18] proposed an energy-efficient resource-provisioning framework with dynamic virtual machine placement using an energy aware load balancer in the cloud. The presented technique obtains measurable improvements in resource utilization by minimizing the power consumption, cost, and execution time of the cloud computing environment.

The proposed scheduling scheme in [19] utilized the resources and saved energy consumption by optimizing the VM allocation and using a consolidation policy. The results showed energy consumption improvement compared with two benchmarks—DVFS and an Energy-aware Scheduling algorithm using Workload-aware Consolidation Technique (ESWCT). The proposed efficient server-first scheduling in [20] used the response time with respect to the number of active servers to reduce the energy consumption in the data center. About 70 times the energy was saved in comparison of the random selection-based strategy.

The previously mentioned task-scheduling algorithms considered one or two parameters of task scheduling, while the algorithm must provide fairness to users when providing services and deliver the ultimate performance with the best reductions in cost and power consumption. Intel's Cloud Computing 2015 Vision stressed the need for dynamic resource scheduling approaches to increase the power efficiency of data centers by shutting down idle servers [21] and using a good task scheduler that can reduce both the power consumption of the utilized resources and the processing time of an application [22].

Following from the limitations of the above studies, this paper focuses on designing a novel scheduler in order to minimize the total processing and communication costs for a set of tasks aiming to reduce the overall energy consumption.

3. Proposed Sharing with Live Migration (SLM) Model

Refining the energy optimization of cloud computing raised the need for optimal solutions and research studies in this area. In this research, we present a novel algorithm for a cloud-computing environment that could allocate resources based on energy optimization methods called SLM. The SLM model takes advantage of two main task characteristics: the similarity among the number of requests delivered to the cloud by the users, and the task migration, in order to reduce the bandwidth utilization and transmission time required for file transactions between different resources and optimize the consumed energy in a data center.

3.1. Workflow Model

To describe the model, the assumption is that there is a set of tasks, and tasks are allowed to be processed on a specific available resource file at the same time if they will perform the same type of process on that source file under specific constrains.

In the process of task scheduling in cloud computing, the environment is as follows:

Inputs: $R = [R_1, R_2, \ldots, R_j, \ldots, R_m]$ is the set of m available resources, which should process n independent tasks denoted by the set $T = [T_1, T_2, \ldots, T_i, \ldots, T_n]$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$.

Outputs: The output is efficient task scheduling to suitable resources and makespan.

Objectives: Improving the energy efficiency of the data center and minimizing makespan to attain energy-efficient scheduling.

The cloud users of the data center send tasks, while the scheduler controls the dependencies between the tasks and handles the distribution process of these requests to the appropriate resources. This paper proposes a novel SLM scheduler that aims to concurrently process multiple tasks in the cloud to optimize the overall energy consumption and makespan without sacrificing the application performance.

3.2. Energy Model

The power model used in the SLM database, as defined in [23], is:

$$P(u) = k^* P_{max} + (1 - k)^* P_{max} * u \quad (1)$$

where P_{max} is the maximum power consumed when the host is fully utilized, k is the fraction of power consumed by the host when idle, and u is the CPU utilization. The CPU utilization is a function of time

that is represented as *u(t)*, so the total energy consumption by one host is calculated as an integral of the power consumption function over a period of time (2).

$$E = \int_t P(u(t)) \tag{2}$$

The sum of energy consumed by the hosts processing all of the tasks at the data center for a given period of time will define the data center energy consumption after executing all of the tasks, terminate VMs, and hosts. The output of the algorithm shows the energy consumption of each host, and the processing time used in the formula to calculate the host energy consumption after executing a single task [23].

3.3. The Execution Model

The following steps discuss the proposed SLM scheduler algorithm:

1. Users send a number of tasks to the data center.
2. The scheduler starts processing tasks in case the needed server is idle.
3. Queuing applied to incoming tasks if their needed file is in use.
4. A task checkup procedure is performed by the scheduler for each of the queued tasks for each queue. In case any of the tasks need to perform a similar type of process on the available resource, it will grant them permission to start concurrently with the current task without waiting in the queue if the designated host is not overloaded.
5. Migrate tasks to the replica underutilized server if the needed main server is overloaded.
6. Migrate tasks from the under-loaded replica server to the main server, then set the replica back to sleep mode.
7. Apply queuing on the tasks with minimum waiting time in case resources are unavailable.

4. SLM Scheduler Algorithm

The SLM algorithm takes advantage of two main attributes: task characteristics similarity and live migration. Similarity refers to processing tasks simultaneously as a set at the same time based on their types. Since we have four types of tasks used in the SLM scheduler (read, write, download, and upload), the scheduler classifies the tasks based on their type and needed source file. For example, if we have two processes that are both reading the same file, they are going to be scheduled concurrently to improve the utilization of the bandwidth, while 'write' and 'write' or 'upload' and 'upload' cannot be processed concurrently due to overwrite conflicts. Live migration can also contribute to power saving, as VM migrations can minimize the number of running physical machines by moving tasks to underutilized machines (consolidation) [24] and moving processes from overloaded to less loaded servers (load balancing) [25]. In general, the typical parameter considered by a cloud provider to facilitate auto-scaling is the threshold. The threshold parameter defines the aggregate value calculated based on current performance metric values, at which point the auto-scaling of resources is triggered, too. The upper and lower thresholds of auto-scaling are set to 80% and 30%, respectively, derive from pre-agreed Service Level Agreements (SLAs) based on [26]. The overload migration starts if the load is higher than 80%, because the value of the upper utilization threshold is 80%. In the SLM model, migration happens between two replica servers, where the SLM data center model is created using the server replica configuration. The replica configuration means that a duplication of each server is used to function as an up-to-date replica of the master server. The following equations were used in the SLM model to calculate the cost of the VMs' migration [27].

$$T_{m_j} = \frac{Memory_j}{Bandwidth_j} \tag{3}$$

$$U_{d_j} = 0.1 * \int_{t_0}^{t_0+T_{m_j}} U_j(t)dt \quad (4)$$

where U_{dj} is the total performance degradation by VM_j, t_0 is the time when the migration starts, T_{mj} is the time taken to complete the migration, U_j (t) is the CPU utilization by VM_j, $Memory_j$ is the amount of memory used by VM_j, and $Bandwidth_j$ is the available network bandwidth.

In order to apply the SLM algorithm, one read/download counter and one write/upload lock are defined and used, as introduced in Algorithm 1, which are identified as read/download lock and write/upload lock, respectively. The write/upload lock is turned on while performing a write or upload process. Once the process is completed, the lock will be switched back to off. This lock is to prevent any other process from starting as long as it is on. For the read/download counter, one counter is used to count the read and download operations, because a write or upload operation on a resource can start only if no other process is using it, since many read or download tasks can be processed in parallel, but do not necessarily start and end at the same time. The counter will be incremented each time a read or download process starts, and decremented each time a read or upload process has finished. The write or upload process cannot start unless this counter is zero, which is why we use this counter.

Algorithm 1: The Sharing with Live Migration (SLM) algorithm

```
Create different types of task
Submit task to broker for execution
For each task from task queue
  For each VM from VM list
      Submit task to the VM
    If VM.host.utilization > 80
        Migrate VMs until host utilization is <80
    Else
      If VM.host.utilization < 30
          Migrate to another host and hibernate current host
        If the read/download counter=0 & write/upload lock is in off mode for the current task required
resource file
        Submit task for execution
            If the task type is read/download
                Increment the read/download counter
            Else
                Set write/upload lock on
    Else
            If read/download counter ≠0
              If task type is read/download
                  Submit task for execution, increment counter
            Else
                Submit task to waiting queue
    Else
        Submit task to the waiting queue
After execution of task
If task has used file for read or download
    Decrement read/download count
Else
    Set write/upload off
```

5. SLM Simulation and Analysis

In the proposed approach of the SLM, we used the Cloud-Sim toolkit to simulate the total energy consumption. Then, we compared the SLM simulation results with the data center design in [28].

5.1. Assumptions

The cloud data center proposes six physical servers, each with 10 virtual machines, and there are several characteristics for all of the VMs and hosts, where each had the exact same specific characteristics, such as processing capability, storage, and bandwidth. Each host has eight GB of RAM, 2 TB of storage, four CPUs that have a capacity power of 10,000 MIPS, and the data center is 16 DVS-enabled processors [29]. Each host has three different source files required by the cloud users, and since the data center has six hosts, we applied the replica method on half of the hosts, which means that three hosts have nine different sources, while the other half of the hosts have the exact copy of the nine files, as shown in Figure 1.

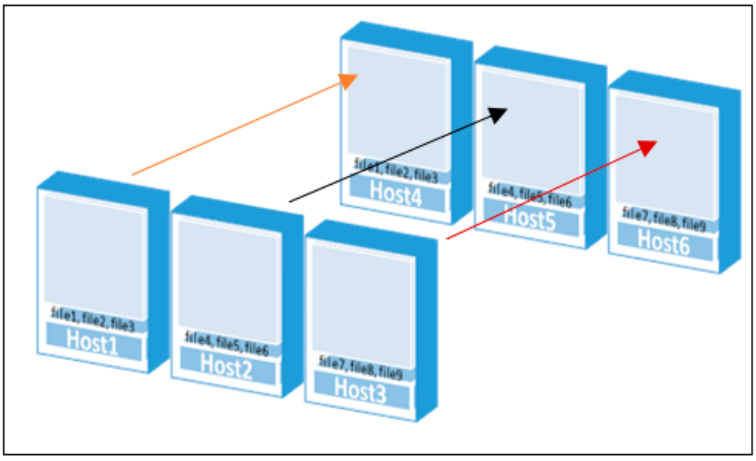

Figure 1. Identical hosts.

5.2. Simulation Environment

We used Cloud-Sim as the simulation platform, as it enables modeling and simulating the cloud computing systems, provides a wide range of application environments, and supports both the system and behavior modeling of cloud systems, such as data centers, virtual machines, and resource provisioning policies [29].

The aim of the simulation is to evaluate the efficiency of the algorithm in a typical data center scenario. In order to simulate the datacenter, six hosts and 60 VMs were included, as well as 500 tasks requiring four different types of processing (reading, downloading, writing, or uploading), and nine source files located in hosts to be processed by the different tasks.

5.3. Simulation Scenario

A broker entity connects users to the cloud. First, the user submits a task to its broker; then, the broker schedules the tasks based on a scheduling policy. Before scheduling the task, the broker dynamically gets a list of available resources and information regarding the hosts where they are located. The simulated scenario includes the generation of 500 tasks, which are passed to a scheduler to control and prioritize their admission. The simulation environment had been set up and executed following the next steps in the scenario:

(1) Generating tasks: Set up task types to reading, writing, uploading, and downloading with task lengths of 10,000, 40,000, 80,000, and 200,000. Generate a random number of tasks and set their configuration (type, size, number).

(2) Primary admission control: Data center broker submits tasks to a class called 'cloudlet' to manage transferring it to the proper VM without violating any concurrency control rule while there are enough free processing elements (PEs).

(3) Application-specific analysis and ranking: The file execution details describe each task, such as file VM number, file host number, file number, and task type. The SLM scheduling policy assign tasks directly to the first under-loaded VM that has the corresponding required file. The 'cloudlet' type represents the type of the task process, which could be either reading, downloading, writing, or uploading, each bearing a corresponding code. The SLM scheduler allows multiple tasks processing at the same time only if the task's type is 'read' or 'download' (not if the current task type under processing is either 'write' or 'upload', due to the overwriting problem). Hence, the SLM scheduler will provide concurrency control.

(4) QoS and SLA control: The proposed SLM simulation model added a continuous observation class to the simulation to detect changes of the QoS and the SLA conditions, and prevent any violation to those agreements.

6. Experimental Results

In this section, we analyze the performance of our algorithm based on the experimental results. We created a simulation named SLM to further verify the performance of the proposed SLM scheduler. To test the response time performance of our proposed SLM scheduler, we benchmarked it against a scenario named basic work [30]. The basic work model is a greedy data center that assigns tasks to the hosts with sufficient resources to handle the tasks using a FIFO (First In First Out) queuing system and no concurrency control, where two tasks cannot access the same file simultaneously at the same time. Therefore, at high loads, the waiting and response time will eventually increase the makespan.

Two parameters have been used to measure the performance of the task scheduling algorithms: energy consumption and makespan, where makespan is the overall completion time needed to assign all of the tasks to the available resources i.e., VMs. Hence, the SLM model will provide task admission control for all of the incoming tasks, and it will provide a parallel processing feature with migration. Compared with the basic work algorithm, the proposed algorithm can significantly reduce the system energy consumption and makespan, as shown in Table 1.

Table 1. The SLM and the basic work simulation energy and makespan results.

No. of Tasks	Basic Work Model		Energy Consumption in the SLM Model	
	Energy Consumption (kwh)	Makespan (s)	Energy Consumption (kwh)	Makespan (s)
100	0.16	1098	0.07	450
200	0.24	1590	0.11	730.1
300	0.3	2178	0.17	1260
400	0.32	2290	0.21	1350.1
500	0.39	2518	0.27	1920

The simulation results of the SLM model in Table 1 show the improvement in reducing energy consumption. If we take an example of a batch of 100 tasks in the cloud data center, those tasks are processed using 0.16 kWh and 1098 s of makespan in a basic work environment, while in the SLM model, they require only 0.07 kWh and 450 s of makespan. Figures 2 and 3 display the improvements.

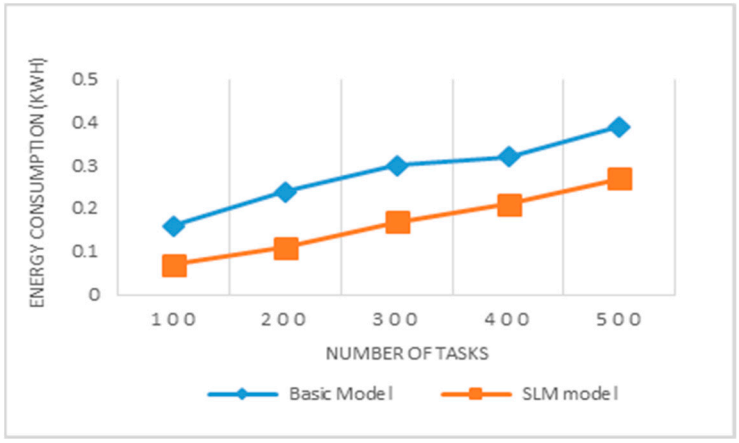

Figure 2. Comparison of the energy consumption of the SLM and the basic work model.

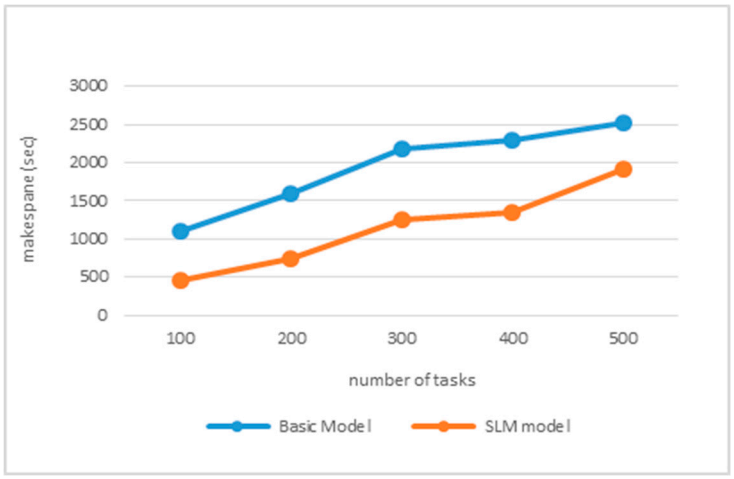

Figure 3. Comparison of the energy consumption of the SLM and the basic work model.

7. Conclusions

The ultimate goal of energy-efficient scheduling in cloud computing is to reduce the costs and computing infrastructure. This paper introduced a novel scheduling algorithm, named SLM, which aims to reduce the overall energy consumption of a data center by combining sharing and live migration policies. A set of simulation tests demonstrated that the method leads to a more balanced workload for each machine. Compared to a basic work, the SLM algorithm has a shorter processing time and an overall optimized performance. The results show good performance in energy utilization and makespan. We expect that the real-world cloud platforms will apply the proposed algorithm, aiming at network load minimization and reducing the energy costs for cloud data centers. In the future, we plan to extend the work for 500 to 10,000 tasks in order to evaluate energy consumption.

Author Contributions: Conceptualization and Methodology, S.A., B.G., N.C.; Investigation, S.A. and B.G.; Software, Validation, Formal Analysis, X.X.; Writing-Original Draft Preparation, S.A.; Writing-Review & Editing, S.A., B.G., N.C.; Supervision, B.G., N.C.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, S.; Jeong, Y.; Hyuk, J. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222. [CrossRef]

2. Azzouzi, M.; Neri, F. An Introduction to the Special Issue on Advanced Control of Energy Systems. *WSEAS Trans. Power Syst.* **2013**, *8*, 103.
3. Singh, S.; Sharma, P.K.; Moon, S.Y. EH-GC: An Efficient and Secure Architecture of Energy Harvesting Green Cloud Infrastructure. *Sustainability* **2017**, *9*, 673. [CrossRef]
4. Xu, D.; Wang, K. Stochastic Modeling and Analysis with Energy Optimization for Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2014**, *5*, 1–5. [CrossRef]
5. Patel, S.; Makwana, R.M. Optimized Energy Efficient Virtual Machine Placement Algorithm and Techniques for Cloud Data Centers. *J. Comput. Sci.* **2016**, *12*, 1–7. [CrossRef]
6. Baiboz, A. Energy-Efficient Data Center Concepts under the EXPO-2017 Astana. *J. Multidiscip. Eng. Sci. Technol.* **2015**, *2*, 1126–1128.
7. Awad, A.I.; El-Hefnawy, N.A.; Abdelkader, H.M. Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments. *Procedia Comput. Sci.* **2015**, *65*, 920–929. [CrossRef]
8. Mahmood, A.; Khan, S.A.; Bahlool, R.A. Hard Real-Time Task Scheduling in Cloud Computing Using an Adaptive Genetic Algorithm. *Computers* **2017**, *6*, 15. [CrossRef]
9. Ma, T.; Tang, M.; Shen, W.; Jin, Y. Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing. *Information* **2017**, *5*, 25. [CrossRef]
10. Yadav, A.K.; Rathod, S.B. Study of Scheduling Techniques in Cloud Computing Environment. *Int. J. Comput. Trends Technol.* **2015**, *29*, 69–73. [CrossRef]
11. Yadav, A.K.; Mandoria, H.L. Study of Task Scheduling Algorithms in the Cloud Computing Environment: A Review. *Int. J. Comput. Sci. Inf. Technol.* **2017**, *8*, 462–468.
12. Tang, Z.; Qi, L.; Cheng, Z.; Li, K.; Khan, S. An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment. *J. Grid Comput.* **2015**, *14*, 55–74. [CrossRef]
13. Wu, C.; Chang, R.; Chan, H. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Gener. Comput. Syst.* **2014**, *37*, 141–147. [CrossRef]
14. Ali, S.A.; Islamia, J.M. A Relative Study of Task Scheduling Algorithms in Cloud Computing Environment. In Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, India, 14–17 December 2016. [CrossRef]
15. Zhao, Q.; Xiong, C.; Yu, C.; Zhang, C.; Zhao, X. A new energy-aware task scheduling method for data-intensive applications in the cloud. *J. Netw. Comput. Appl.* **2016**, *59*, 14–27. [CrossRef]
16. Lin, C.; Syu, Y.; Chang, C.; Wu, J.; Liu, P.; Cheng, P.; Hsu, W. Energy-efficient task scheduling for multi-core platforms with per-core DVFS. *J. Parallel Distrib. Comput.* **2015**, *86*, 71–81. [CrossRef]
17. Ding, Y.; Qin, X.; Liu, L.; Wang, T. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Future Gener. Comput. Syst.* **2015**, *50*, 62–74. [CrossRef]
18. Pavithra, B.; Ranjana, R. Energy efficient resource provisioning with dynamic VM placement using energy aware load balancer in cloud. In Proceedings of the 2016 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 25–26 February 2016. [CrossRef]
19. AlIsmail, S.M.; Kurdi, H.A. Green algorithm to reduce the energy consumption in cloud computing data centers. In Proceedings of the 2016 SAI Computing Conference (SAI), London, UK, 13–15 July 2016; pp. 557–561.
20. Ziqian, D.; Liu, N.; Rojas-Cessa, R. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *J. Cloud Comput. Adv. Syst. Appl.* **2015**, *4*, 5. [CrossRef]
21. Intel's Cloud Computing 2015 Vision. Available online: http://www.intel.com/content/www/us/en/cloud-computing/cloudcomputing-intel-cloud-2015-vision.html (accessed on 25 July 2018).
22. Ismaila, L.; Fardoun, A. EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems. In Proceedings of the 6th International Conference on Sustainable Energy Information Technology (SEIT 2016), Madrid, Spain, 23–26 May 2016; pp. 870–877.
23. Tian, W.; Xu, M.; Chen, A.; Li, G.; Wang, X.; Chen, Y. Open-Source Simulators for Cloud Computing: Comparative Study and Challenging Issues. *Simul. Model. Pract. Theory* **2015**, *1*, 239–254. [CrossRef]
24. Kaur, P.; Rani, A. Virtual Machine Migration in Cloud Computing. *Int. J. Grid Distrib. Comput.* **2015**, *8*, 337–342. [CrossRef]
25. Alshathri, S. Towards an Energy Optimization Framework for Cloud Computing Data Centers. In Proceedings of the Eleventh International Network Conference (INC 2016), Frankfurt am Main, Germany, 19–21 July 2016; pp. 9–12.

26. Beloglazov, A. Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing. Ph.D. Thesis, The University of Melbourne, Melbourne, Australia, February 2013.
27. Beloglazov, A.; Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurr. Comput. Pract. Exp.* **2012**, *24*, 1397–1420. [CrossRef]
28. Shu, W.; Wang, W.; Wang, Y. A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2014**, *2014*, 64. [CrossRef]
29. Han, G.; Que, W.; Jia, G.; Shu, L. An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing. *Sensors* **2016**, *16*, 246. [CrossRef] [PubMed]
30. Github. Available online: https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3 (accessed on 25 July 2018).